Carnegie Mellon

School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890

(412) 268-2602

10 April 1992

Dr. Susan Chipman
Personnel & Training
RES Code 442 PT
ONR
800 N. Quincy Street
Arlington, VA 22217

Dear Susan:

This letter constitutes the Performance Technical Reports for the four initial quarters of the ASPM project, officially titled *Fitting Symbolic Parameter Cogntive Models*, with contract number N00014-91-J-1527.

The initial version of ASPM (ASPM0), which was created prior to the grant, was written in Lisp and was tied to the analysis of a particular computational model (of syllogistic reasoning). The main activity during the first three quarters has been to build a version of ASPM (ASPM1) written in C, which is approximately 30 times faster and is not tied to any particular model. We demonstrated ASPM1 at the Cognitive Diagnosis Contractor's Meeting in Pittsburgh on 11-12 Oct 91. We distributed at the meeting a preliminary version of a paper on ASPM, T. Polk, A. Newell and K. VanLehn, *Analysis of Symbolic Parameter Models: A new technique for fitting models in cognitive science.* It is preliminary because we want to add a second example (Deb) and bring everything to being done with ASPM1, rather than ASPM0. We will then submit it for publication.

Since Oct 91 we have been using ASPM1. We have successfully applied it to a new syllogism model that has nearly 11 billion parameter settings (compared with 0.7 billion settings in the previous model). Despite the enormous size of this parameter space, the system was able to compute the entire range of model behaviors on 16 tasks in about 4 hours (this is the *compose* operation described in the ASPM papers), and then to find all optimal settings for an individual subject in about 30 minutes (the *fractionate* and *search* operations). (Times are on a DecStation 5000/200.)

We have extended the system to allow the analysis of arbitrary subsets of parameter space. This functionality makes possible some interesting new analyses. For example, previously ASPM simply produced the optimal settings and the user would have to recover the model's predictions by running the model using those settings. Now ASPM allows the user to restrict the analysis to that subset of parameter space consisting of the optimal settings. The model predictions can then be computed by using the *compose* operation on this subset of parameter space to compute the range of model behaviors. This operation is computationally cheap because the number of optimal settings is small compared to the number of possible settings. So by applying ASPM to its own results,

the user can recover the range of behaviors predicted by the optimal settings. This type of analysis is important in figuring out where a model is breaking down and how it might be improved. This extension is another vindication of our expectation that ASPM provides an entire suite of analysis tools.

ASPM1 is now available for use by other researchers and we are beginning collaborations. Work on Deb, the subtraction model that Kurt VanLehn and Anandeep Pannu have recently built in Prolog, is essentially ready to be put through ASPM as of the end of this reporting period (15 Feb 92). In addition, some collaborations within the research community surrounding the Soar project (with Richard Lewis at CMU and Jim Herbsleb at Michigan) are in the planning stage. Nevertheless, there are still some obstacles to the widespread use of ASPM, which we need to overcome in future work. Perhaps the most serious obstacle is the fact that ASPM can not yet interface directly with users' computational models. ASPM requires complete information about a model's behavior on every task (a *primitive response partition* consisting of the entire range of model behaviors on that task and the settings that lead to each behavior). In the present system, this information must be given to ASPM in a specified syntax. It is unreasonable to expect naive users to generate this information themselves. Instead, ASPM should compute this information itself by running the user's model. To do so ASPM must be able to communicate with and execute programs written in programming languages other than C (e.g., Lisp, Prolog...). We plan to start working on such an ability in the near future. Other essential improvements for usability, such as a high-quality computer interface for ASPM and interfacing with a set of statistical tools, are further in the future.

Sincerely,

Allen Newell

Allen Newell
U. A. and Helen Whitaker University Professor
of Computer Science

AN:mp

cc: Office of Sponsored Research, CMU
Charles E. Davis, ONR
Kathy Williams, ONR Resident Rep
Administrative Grants Officer, ONR
Director, Naval Research Lab
Defense Technical Information Center ✓